5. Coding for Efficiency

Suppose we have a very long message which we want to save or send to someone, and we would like to shorten it to save ourselves space or sending time. This happens in particular if our message contains some kind of audio or video information, which can go on and on.

There are all sorts of procedures and protocols used to compactify information. We will study one simple problem here, and maybe another..

Suppose we know nothing whatever about our message, and want to compress it using only information we can glean from it by making a relatively small amount of counting on its data.

Suppose then that our message is represented as a long string of 0's and 1's of length L. One thing we can do is to pick a small number, k, divide the message up into consecutive blocks each consisting of k bits of information, (that is, k digits) and count how many blocks there are in our message for each possible block pattern of length k.

There are, of course, 2^k different possible patterns of digits 0 and 1 of length k.

If we were to do this, we would find, say, f(q) patterns of form q for each such pattern q.

If k is 2, the possible patterns are 00, 01, 10 and 11, which we can call 0,1,2 and 3 (which corresponds to reading them in binary notation). So we will get four frequencies, f(0), ..., f(3). Their sum will be N/k or rather the smallest integer at least as large as N/k, since that is how many blocks we will have.

So we could find 2^k

such pattern occurrence numbers, (we call them pattern frequencies) and we want to use this information **to shorten our message**.

Note: Here we have created *words*

that are binary sequences of length k. What we do below applies equally well when words are defined in any way at all, so long as they have given frequencies of occurrence.

How can we do so?

Well, we can assign a *code word*

to each pattern, and make the code words that correspond to our frequently occurring patterns short, and those corresponding to rare patterns long.

This is what Morse code does for English language text, with patterns being individual letters.

This brings us to two specific questions:

1. **How much can we shorten our message by using this technique?**
2. **How can we find an efficient shortening algorithm for this problem?**

We will address these in turn,

5.1 Shannon's First Theorem: A Lower Bound to the Length of the Shortened Message

We now look at the first of these questions:

What can we say about the length of the shortest possible message that is a sequence of binary bits, based only on pattern frequencies?

We employ our old friend **the Pigeonhole Principle**, to get a bound.

To apply that principle, we must address two questions:

If we shorten our message so it ends up being M bits long, how many different possibilities can we distinguish?

How many arrangements of our patterns are there, consistent with our pattern frequency data?

The first question is straightforward: if our shortened message is M bits long, there are 2^M possible messages that can be distinguished by them.

So let us look at the second question:

We know from our frequency information how often each pattern occurs; but we have no information about the ordering of the patterns in our message. Our message can be any ordering of f(q) samples of block q for each q.

We let the total number of patterns, which is the sum over q of f(q), be denoted by **N**.

The answer to the second question is what is called a ***Multinomial Coefficient.***

You have undoubtedly run into the most common of these, which occurs when there are only two patterns. Then there will be j occurrences of the first pattern, say, and N-j occurrences of the second. The number of possible arrangements of these is **the *binomial coefficient C(N,j)***, (It is often written as an N atop a j within brackets. This requires a minor effort to type here so we use the C notation)

And what is *C(N,j)*
explicitly? One way to compute it is to count the number of ordered arrangements, and divide that by the number of ordered arrangements that correspond to the same unordered arrangement.

The number of ways of arranging N patterns is N! (Any of N can go first, any of N-1 can go next, etc.no matter what the previous choices were.) Furthermore, rearranging any of the j copies of the first pattern will not change the unordered pattern,
no will rearranging any of the N-j copies of the second pattern. On the other hand, any other rearrangements will alter the pattern.

We conclude:

$$\mathbf{C(N,j) = N!/j!(N-j)!.}$$

And what happens with more than two patterns?

If there are p distinct patterns the notation is ***C(f(1),f(2), ..., f(p))***. (Notice that with the binomial coefficient we suppress the second pattern frequency, while usually in general we do not do so.)

By the identical argument given for binomial coefficients we find the following explicit formula for multinomial coefficients:

$$\mathbf{C(N, f(1),f(2),...,f(p)) = N!/(f(1)!\ f(2)!\ ...\ f(p)!).}$$

We can now apply our pigeonhole principle, which tells us we must have

$$2^M \geq \mathbf{C(N, f(1),f(2),...,f(p))}$$

if
there are to be enough different messages to allow us to distinguish among all possible of their orderings.

We take logs to the base 2 of both sides of this inequality to obtain:

$$\mathbf{M \geq \log_2 C(N, f(1),f(2),...,f(p)).}$$

To deduce the implications of this inequality we need an expression for factorials that is easy to manipulate algebraically. Fortunately there is a very nice one, known as **Stirling's approximation.** In its crudest form, which as all we need here, it reads:

$$R! = (R/e)^R * W(R),$$

where W(R) is at least 2 and is no greater than cR for some small constant c.

We will prove this soon**,** and you will actually prove a very accurate approximation formula for W(R)**.** Notice that the expression here implies

$$\mathbf{Log_2\ R! = Rlog_2(R/e) + O(log_2\ R)}$$

The right side of our (log) pigeonhole inequality above involves lots of logs of factorials**.** It is convenient for us to express **f(j), the frequency of the j-th pattern,** as **$N*p_j$, where p_j is then the relative frequency of that pattern by which mean the proportion of the words in the message that are of that pattern.**

It is useful to look at our lower bound on the length of the compressed message in the case that our original message itself had only two patterns, for example,. 0 and 1.
We then find

$$M \geq \log_2 C(N, Np_1)$$

$$\geq N\log_2(N/e) - Np_1\log_2 (Np_1/e) - N(1-p_1)\log_2(N(1-p_1)/e) + O(\log N)$$

By using the fact that the log of a product is the sum of their logs, we can separate the p terms in the logs here from the rest, and find that the non-p terms cancel out, and we get

$$\mathbf{M \geq N(-p_1log_2(p_1) - (1-p_1)log_2(1-p_1))}$$

Claude Shannon called the expression on the right here $NH(p_1)$ and called it the Entropy of Information of the original sequence.
It is the number of binary bits needed to describe the original message. In the case of more patterns, it looks just the same, as you should verify by writing out all the factorials and logs and simplifying yourself. The result is

$$\mathbf{H(p(1) \ldots p(p)) = \ - \Sigma(from\ j=1\ to\ j=p)\ of\ p_j\ \ log_2\ \ p_j}$$

With this notation we obtain the following result:
Theorem: Suppose we compress a sequence consisting of N words that are each instances of one of p patterns. If we only use frequency information contained in the frequencies f(s) of the various patterns, we will need a number of binary digits for our message that is at least

$$\mathbf{N*H(\{p_0, p_1, \ldots, p_p\})}$$

and we can can find a way to compress our message in this way that uses at most

$$\mathbf{N*(H(p_0, p_1, \ldots, p_{N/(k-1)})+1}$$

binary bits. (Usually we get much closer to the former number.)

By writing out the pigeonhole principle bound, applying the Stirling formula throughout and taking logs you can prove the first statement of this theorem. The second statement, that you can come within N of this

bound, which is one extra bit per word of the message, follows from two observations.

The actual length of the encoded message will be the sum over all j of the number of occurrences of the j-th pattern in it multiplied by the length of the code word assigned to that pattern. If the length of the j-th code word is $\log_2 p(j)$ then the Shannon bound is exactly achieved.

First, if each p(j) here is the reciprocal of a power of 2, then the bound can be achieved exactly. This can be done by assigning a binary sequence of length $-\log_2 p(j)$ as code word for the j-th pattern.

Second, you can in any case form a code by assigning a binary sequence of length the smallest integer greater than $-\log_2 p_j$ to the j-th pattern, and that will be at most one greater than $-\log_2 p_j$ itself. This each encoded word will be at most one bit longer than the corresponding term in the Shannon bound.

As a matter of fact you can improve this result by replacing the extra 1 bit per word here by $2^{-c+1}$where c is the minimum over j of the smallest integer greater than $-\log_2 p_j$, which would be the length of the smallest code word.

A binary tree is a tree in which each vertex has either 2 or 0 descendants. We can assign a code word to each leaf of a binary tree by starting at the top and assigning 0 to one descendant and 1 to the other, adding the bit onto the bits already possessed by the parent. If we do this the words at depth d in the tree will have a code word of length d. Moreover, if we sum 2^{-d}
over all leaves of the tree, we get 1. (Prove these statements to yourself).

Since the p_j
sum to 1 as well, when they are inverse powers of 2 we can construct a tree with a leaf of depth given by $\log_2 p_j$ for each j, and the corresponding code will assign a code word of that length to the j-th pattern. And that is all there is to the first observation. The second is trivial.

We now ask, why do we call H here entropy of information?.

5.2 What is Entropy? What is Entropy of Information?

The physical concept of entropy was developed in the subjects of thermodynamics and statistical mechanics.

If we look at a small piece of a large interacting system, we will find that it will not in general be in some fixed state. Rather, it can be found in any one of a usually large number of different states. If our small piece exchanges energy with the rest of the system, we will find it in states of differing energy at different times.

Its entropy is a logarithmic measure of the diversity of the states of the system that can be observed in our small subsystem. These states generally appear with a frequency proportional to a factor of exp(-E/kT) where E represents the energy of the state, and T is the temperature of the system. In physics the entropy is proportional to the logarithm of the sum over all states of the exponential factor just mentioned.
At high temperatures the exponential factor is close to 1 for many states and the entropy is high. At low temperature, only states with close to the lowest possible energy occur much at all, and the entropy is usually very low.
There are two laws of thermodynamics that involve entropy. The second law states that entropy of a system left on its own never decreases. That reflects the physical fact that states do not coalesce spontaneously, and a diverse system will retain its diversity.
There is also a third law of thermodynamics which states that the entropy of a physical system approaches 0 near 0 temperature. This is of course only true of systems with very few low energy states. At low

temperature only these states predominate in our subsystem and there is little possible diversity among the states .There is an exception to this law, and it is the substance known as water, but that is beyond our present horizon.

Shannon
invented the concept of entropy of information, as a logarithmic measure of the diversity among potential messages that is wiped out upon receipt of a specific message. It thus provides a measure of how much information, how many bits you actually learn when you get a message.
In the case of information considered here, if we know the frequency of our patterns, the diversity of potential messages comes entirely from the different orders in which these blocks can occur.
In our case, our frequency information tells us that the number of potential messages is a certain multinomial coefficient. The log to the base 2 of this coefficient is number of bits you must have in order to exceed the same number of potential messages, and is the number of bits you need supply in order to reduce the original diversity down to one particular message.

5.3 Stirling's Formula and Approximations for Multinomial Coefficients

We have written a formula for a multinomial coefficient in terms of factorials, so we can write it and its logarithm conveniently if we can find a convenient approximation for n!. We have claimed that there is such a formula and it can be written as follows:

$$\mathbf{n! = (n/e)^n\ W,}$$

where W is a factor on the order of no more than n (and actually $n^{1/2}$).

In fact we can write $\mathbf{W= (2\ \pi\ n +\pi/3)^{1/2} \cdot(1 + O(1/n)).}$

When we take logarithms of n! the
factor W recedes into insignificance, and contributes nothing to the leading terms here. It can therefore be ignored in our present discussion.

(It is curious that the approximate formula implied above for n! is quite accurate even for n=0, (if you interpret $(0)^0$ as 1) and gets better as n increases in that the ratio the formula to n! approaches 1 as n increases..)

An argument for this form of approximation comes from the power series expression for the function exp(n).

This series is:

$$\mathbf{Exp(n) = 1 + n + \ldots\ n^{n-1}/(n-1)! + n^n/n! + n^{n+1}/(n+1)! + \ldots}$$

The ratio of the term $n^j/j!$ to $n^{j-1}/(j-1)!$ is n/j. Thus the terms increase up to $n^{n-1}/(n-1)!$, then the next term is the same and then they start downward, at ever increasing speeds since the ratio of the kth term to the (k-1)st is n/k.

We can therefore represent the left-hand-side, the exponential function, by the product of the largest term on the right and a measure of how many terms on the right make major contributions, in short by $n^n/n!$ *W where W is easily seen to be less than n.

And we can rearrange this relation to read n! = W*n^n/exp(n) as claimed, with 1< W < n.
The notation O(f(n)) means that, as n goes to infinity, this quantity behaves like f(n), in that its ratio to f(n) approaches at most a constant, while o(f(n)) means its ratio to f(n) goes to 0 as n increases.
Exercises:

1. Use a spreadsheet or other means to plot the function H(p1) in the range 0 to 1 for p1.

2. On a spreadsheet or otherwise compute $\mathbf{n!/(n/e)^n\ /n^{1/2}}$ **for n = 2,4,8,16,32,64,128**
Use these results and extrapolation to estimate the limit of this ratio.
(to extrapolate look at **s(k) = 2r(2k) − r(k)**
then **t(k) = 4s(2k)/3 − s(k)/3**
then **u(k) = 8t(2k)/7 − t(k)/7**
then **v(k) = 16u(2k)/15 − u(k)/15**
Compare your answer to $\mathbf{(2\ \pi)^{1/2}}$**.**

3. Given the following frequencies f(q) for q=0 to 15, compute the entropy of this list (compute their total, n, the p(q) and then the entropy function of these)
1, 22,11,15,2,1,2,3,45,95,2,2,1,2,25,1